

Machine Learning

Bayesian Learning

Artificial Intelligence & Computer Vision Lab
School of Computer Science and Engineering
Seoul National University

Overview

- Introduction
- Bayes Theorem
- Review of other learning methods by Bayesian approach
- Bayes Optimal Classifier
- Gibbs Algorithm
- Naive Bayes Classifier
- Example : Learning to Classify Text
- Bayesian Belief Networks
- The EM Algorithm

Introduction

- Bayesian learning methods are relevant to our study of machine learning.
 - Bayesian learning algorithms are among the most practical approaches to certain types of learning problems
 - ex) the naive Bayes classifier
 - Bayesian methods provide a useful perspective for understanding many learning algorithms that do not explicitly manipulate probabilities

Introduction (cont.)

- Features of Bayesian learning methods
 - Each observed training example can incrementally decrease or increase the estimated probability that a hypothesis is correct (flexible)
 - Prior knowledge can be combined with observed data to determine the final probability of a hypothesis
 - Bayesian methods can accommodate hypotheses that make probabilistic predictions
 - New instances can be classified by combining the predictions of multiple hypotheses, weighted by their probabilities

Introduction (cont.)

- Practical difficulties
 - Require initial knowledge of many probabilities
 - The significant computational cost required to determine the Bayes optimal hypothesis in the general case

Bayes Theorem

- Determine the best hypothesis from some space H , given the observed training data D
= the most probable hypothesis
 - given data D + any initial knowledge about the prior probabilities of the various hypotheses in H
 - provides a way to calculate the probability of a hypothesis based on its prior probability
- Notation
 - $P(h)$: *initial probability* that hypothesis h holds
 - $P(D)$: *prior probability* that training data D will be observed
 - $P(D|h)$: probability of observing data D given some world in which hypothesis h holds
 - $P(h|D)$: *posterior probability* of h that h holds given the observed training data D

Bayes Theorem (cont.)

- Bayes theorem:

$$P(h | D) = \frac{P(D | h)P(h)}{P(D)}$$

- Is the cornerstone of Bayesian learning methods because it provides a way to calculate the posterior probability $P(h/D)$, from the prior probability $P(h)$, together with $P(D)$ and $P(D|h)$
- It can be applied equally well to any set H of mutually exclusive propositions whose probabilities sum to one

Bayes Theorem (cont.)

- Maximum a posteriori (*MAP*) hypothesis
 - The most probable hypothesis $h \in H$ given the observed data D

$$\begin{aligned}h_{MAP} &\equiv \arg \max_{h \in H} P(h | D) \\ &= \arg \max_{h \in H} \frac{P(D | h)P(h)}{P(D)} \\ &= \arg \max_{h \in H} P(D | h)P(h)\end{aligned}$$

Bayes Theorem (cont.)

- Maximum likelihood (ML) hypothesis
 - assume : every hypothesis in H is equally probable a priori
($P(h_i) = P(h_j)$ for all h_i and h_j in H)

$$h_{ML} \equiv \arg \max_{h \in H} P(D | h)$$

Bayes Theorem (cont.)

- Example: Medical diagnosis problem
 - Two alternative hypothesis
 - (1) the patient has a cancer
 - (2) the patient does not
 - Two possible test outcomes
 - (1) + (positive)
 - (2) – (negative)
 - Prior knowledge
 - $P(\text{cancer})=0.008$ $P(\neg \text{cancer})=0.992$
 - $P(+|\text{cancer})=0.98$ $P(-|\text{cancer})=0.02$
 - $P(+|\neg \text{cancer})=0.03$ $P(-|\neg \text{cancer})=0.97$

Bayes Theorem (cont.)

- Example: Medical diagnosis problem (cont.)
 - Suppose a new patient for whom the lab test returns a positive result

$$P(+|cancer) P(cancer)=0.0078$$

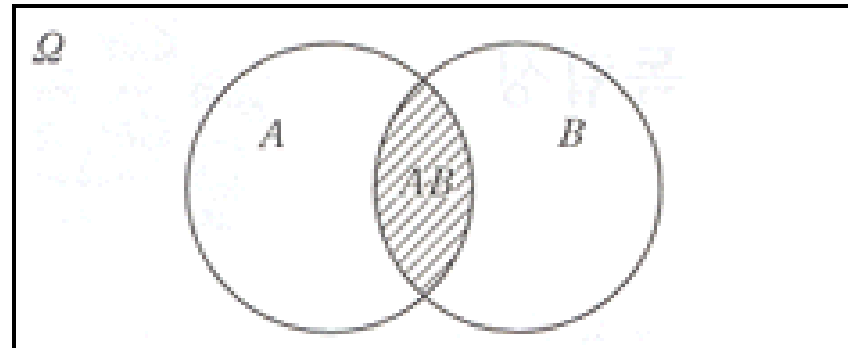
$$P(+|\neg cancer) P(\neg cancer)=0.0298$$

$$\text{thus, } h_{MAP} = \neg cancer$$

$$\begin{aligned} \text{by normalizing, } P(cancer|+) &= \frac{.0078}{.0078 + .0298} \\ &= 0.21 \end{aligned}$$

Bayes Theorem (cont.)

- Basic probability formulas
 - Product rule :
$$P(A \wedge B) = P(A|B) P(B) = P(B|A) P(A)$$
 - Sum rule :
$$P(A \vee B) = P(A) + P(B) - P(A \wedge B)$$



Bayes Theorem (cont.)

- Basic probability formulas (cont.)

- Bayes theorem :

$$P(h|D) = P(D|h)P(h)/P(D)$$

- Theorem of total probability :

If events A_1, \dots, A_n are mutually exclusive with $\sum_{i=1}^n P(A_i) = 1$

then,
$$P(B) = \sum_{i=1}^n P(B | A_i)P(A_i)$$

Review of Other Learning Methods by Bayesian Approach

- Bayesian theorem provides a principled way to calculate the posterior probability of each hypothesis given the training data.
- we can use it as the basis for a straightforward learning algorithm that calculates the probability for each possible hypothesis, then outputs the most probable
- Contents
 - Version space in concept learning
 - Least-squared error hypotheses (case of continuous-value target function)
 - Minimized cross entropy hypotheses (case of probabilistic output target function)
 - Minimum description length hypotheses

Version Space in Concept Learning

- Finite hypothesis space H defined over the instance space X ,
Learn target concept $c : X \rightarrow \{0,1\}$

Sequence of training example : $\langle \langle x_1, d_1 \rangle \dots \langle x_m, d_m \rangle \rangle$, where $d_i = c(x_i)$

1. Calculate the posterior probability

$$P(h | D) = P(D | h)P(h) / P(D)$$

2. Output the hypothesis h_{MAP} with the highest posterior probability

$$h_{MAP} \equiv \arg \max_{h \in H} P(h | D)$$

Version Space in Concept Learning (cont.)

- Assumptions :

- 1) The training data D is noise free
- 2) The target concept c is contained in the hypothesis space H
- 3) No priori reason to believe that any hypothesis is more probable than any other

- $P(h)$

Assign the same prior probability (from 3)

These prior probabilities sum to 1 (from 2)

$$P(h) = \frac{1}{|H|}, \text{ for all } h \text{ in } H$$

Version Space in Concept Learning (cont.)

- $$P(D | h) = \begin{cases} 1 & \text{if } d_i = h(x_i) \text{ for all } d_i \text{ in } D \\ 0 & \text{otherwise} \end{cases}$$

- The posterior probability

$$\left\{ \begin{array}{l} P(h | D) = \frac{0 \cdot P(h)}{P(D)} \quad \text{if } h \text{ is inconsistent with } D \\ P(h | D) = \frac{1 \cdot \frac{1}{|H|}}{P(D)} = \frac{1 \cdot \frac{1}{|H|}}{\frac{|VS_{H,D}|}{|H|}} = \frac{1}{|VS_{H,D}|} \quad \text{if } h \text{ is consistent with } D \end{array} \right.$$

Version Space in Concept Learning (cont.)

- $VS_{H,D}$ is the version space of H with respect to D
- Alternatively, derive $P(D)$ from the theorem of total probability

$$\begin{aligned} P(D) &= \sum_{h_i \in H} P(D | h_i) P(h_i) \\ &= \sum_{h_i \in VS_{H,D}} 1 \cdot \frac{1}{|H|} + \sum_{h_i \notin VS_{H,D}} 0 \cdot \frac{1}{|H|} = \sum_{h_i \in VS_{H,D}} 1 \cdot \frac{1}{|H|} \\ &= \frac{|VS_{H,D}|}{|H|} \end{aligned}$$

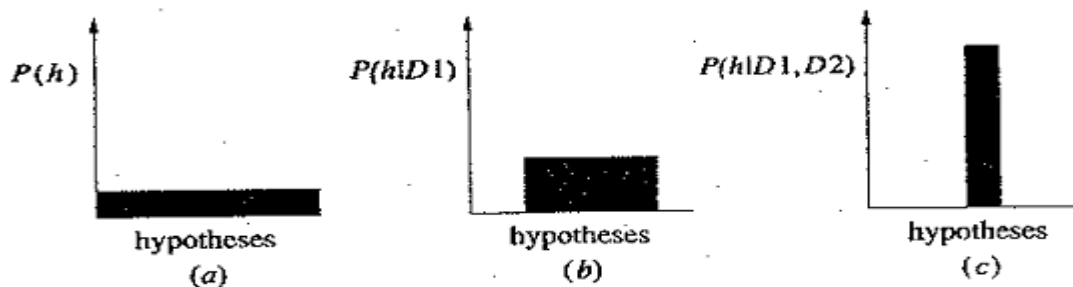
- The hypotheses are mutually exclusive

Version Space in Concept Learning (cont.)

- To summarize,

$$P(h | D) = \begin{cases} \frac{1}{|VS_{H,D}|} & \text{if } d_i = h(x_i) \text{ for all } d_i \text{ in } D \\ 0 & \text{otherwise} \end{cases}$$

- The posterior probability for inconsistent hypothesis becomes zero while the total probability summing to one is shared equally by the remaining consistent hypotheses in $VS_{H,D}$, each of which is a *MAP* hypothesis



Version Space in Concept Learning (cont.)

- Consistent learner, *a learning algorithm which outputs a hypothesis that commits zero errors over the training examples*, outputs a *MAP* hypothesis if we assume a uniform prior probability distribution over H and if we assume deterministic, noise free data (i.e., $P(D|h)=1$ if D and h are consistent, and 0 otherwise 0)
- Bayesian framework allows one way to characterize the behavior of learning algorithms, even when the learning algorithm does not explicitly manipulate probabilities.

Least-Squared Error Hypotheses for a Continuous-Valued Target Function

- Let $f: X \rightarrow \mathbb{R}$ where \mathbb{R} is a set of reals. The problem is to find h to approximate f . Each training example is $\langle x_i, d_i \rangle$ where $d_i = f(x_i) + e_i$ and random noise e_i has a normal distribution with zero mean and variance σ^2

- Probability density function for continuous variable

$$p(x_0) \equiv \lim_{\varepsilon \rightarrow 0} \frac{1}{\varepsilon} P(x_0 \leq x \leq x_0 + \varepsilon)$$

- d_i has a normal density function with mean $\mu = f(x_i)$ and variance σ^2

$$p(d_i) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2\sigma^2}(d_i - f(x_i))^2}$$

Least-Squared Error Hypotheses for a Continuous-Valued Target Function (cont.)

- Use lower case p to refer to the probability density

$$h_{ML} = \arg \max_{h \in H} p(D | h)$$

- Assume the training examples are mutually independent given h ,

$$h_{ML} = \arg \max_{h \in H} \prod_{i=1}^m p(d_i | h)$$

- $p(d_i|h)$ is a normal distribution with variance σ^2 and mean

$$\mu = f(x_i) = h(x_i)$$

$$h_{ML} = \arg \max_{h \in H} \prod_{i=1}^m \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2\sigma^2}(d_i - h(x_i))^2}$$

Least-Squared Error Hypotheses for a Continuous-Valued Target Function (cont.)

- Since $\ln p$ is a monotonic function of p ,

$$h_{ML} = \arg \max_{h \in H} \sum_{i=1}^m \ln \frac{1}{\sqrt{2\pi\sigma^2}} - \frac{1}{2\sigma^2} (d_i - h(x_i))^2$$

$$h_{ML} = \arg \max_{h \in H} \sum_{i=1}^m -\frac{1}{2\sigma^2} (d_i - h(x_i))^2$$

$$h_{ML} = \arg \min_{h \in H} \sum_{i=1}^m \frac{1}{2\sigma^2} (d_i - h(x_i))^2$$

$$h_{ML} = \arg \min_{h \in H} \sum_{i=1}^m (d_i - h(x_i))^2$$

Minimizes the sum of the squared errors between the observed training values and the hypothesis predictions

Least-Squared Error Hypotheses for a Continuous-Valued Target Function (cont.)

- Normal distribution to characterize noise
 - allows for mathematically straightforward analysis
 - the smooth, bell-shaped distribution is a good approximation to many types of noise in physical systems
- Some limitations of this problem setting
 - noise only in the target value of the training example
 - does not consider noise in the attributes describing the instances themselves

Minimized Cross Entropy Hypotheses for a Probabilistic Output Target Function

- Given $f: X \rightarrow \{0,1\}$, a target function is defined to be $f' : X \rightarrow [0,1]$ such that $f'(x) = P(f(x)=1)$.
- Then the target function is learned using neural network where a hypothesis h is assumed to approximate f'
 - Collect observed frequencies of 1's and 0's for each possible value of x and train the neural network to output the target frequency for each x
 - Train a neural network directly from the observed training examples of f and derive a maximum likelihood hypothesis, h_{ML} , for f'

$$\text{Let } D = \{ \langle x_1, d_1 \rangle, \dots, \langle x_m, d_m \rangle \}, d_i \in \{0,1\}.$$

Minimized Cross Entropy Hypotheses for a Probabilistic Output Target Function (cont.)

- Treat both x_i and d_i as random variables, and assume that each training example is drawn independently

$$P(D | h) = \prod_{i=1}^m P(x_i, d_i | h) = \prod_{i=1}^m P(d_i | h, x_i) P(x_i)$$

(x_i is independent of h)

$$\begin{cases} P(d_i | h, x_i) = h(x_i) & \text{if } d_i = 1 \\ P(d_i | h, x_i) = (1 - h(x_i)) & \text{if } d_i = 0 \end{cases}$$

$$P(d_i | h, x_i) = h(x_i)^{d_i} (1 - h(x_i))^{1-d_i}$$

$$P(D | h) = \prod_{i=1}^m h(x_i)^{d_i} (1 - h(x_i))^{1-d_i} P(x_i)$$

Minimized Cross Entropy Hypotheses for a Probabilistic Output Target Function (cont.)

- Write an expression for the ML hypothesis

$$h_{ML} = \arg \max_{h \in H} \prod_{i=1}^m h(x_i)^{d_i} (1 - h(x_i))^{1-d_i} P(x_i)$$

last term is a constant independent of h

$$h_{ML} = \arg \max_{h \in H} \prod_{i=1}^m h(x_i)^{d_i} (1 - h(x_i))^{1-d_i}$$

seen as a generalization of Binomial distribution

- Log of Likelihood

$$\begin{aligned} h_{ML} &= \arg \max_{h \in H} \sum_{i=1}^m d_i \ln h(x_i) + (1 - d_i) \ln(1 - h(x_i)) \\ &= \arg \min_{h \in H} \left(- \sum_{i=1}^m (d_i \ln h(x_i) + (1 - d_i) \ln(1 - h(x_i))) \right) \end{aligned}$$

cross entropy

Minimized Cross Entropy Hypotheses for a Probabilistic Output Target Function (cont.)

- How to find h_{ML} ?
 - Gradient Search in a neural net is suggested

Let $G(h, D)$ be the negation of cross entropy, then

$$\begin{aligned}\frac{\partial G(h, D)}{\partial w_{jk}} &= \sum_{i=1}^m \frac{\partial G(h, D)}{\partial h(x_i)} \frac{\partial h(x_i)}{\partial w_{jk}} \\ &= \sum_{i=1}^m \frac{\partial (d_i \ln h(x_i) + (1 - d_i) \ln(1 - h(x_i)))}{\partial h(x_i)} \frac{\partial h(x_i)}{\partial w_{jk}} \\ &= \sum_{i=1}^m \frac{d_i - h(x_i)}{h(x_i)(1 - h(x_i))} \frac{\partial h(x_i)}{\partial w_{jk}}\end{aligned}$$

w_{jk} : weight from input k to unit j

Minimized Cross Entropy Hypotheses for a Probabilistic Output Target Function (cont.)

- Suppose a single layer of sigmoid units

$$\frac{\partial h(x_i)}{\partial w_{jk}} = \sigma'(x_i) x_{ijk} = h(x_i)(1-h(x_i))x_{ijk}$$

where x_{ijk} is the k th input to unit j for i th training example

$$\frac{\partial G(h, D)}{\partial w_{jk}} = \sum_{i=1}^m (d_i - h(x_i)) x_{ijk}$$

- Maximize $P(D|h)$
 - gradient ascent
 - using the weight update rule

$$w_{jk} \leftarrow w_{jk} + \Delta w_{jk}$$

$$\Delta w_{jk} = \eta \sum_{i=1}^m (d_i - h(x_i)) x_{ijk}$$

Minimized Cross Entropy Hypotheses for a Probabilistic Output Target Function (cont.)

- Compare it to backpropagation update rule, minimizing sum of squared errors, using our current notation

$$w_{jk} \leftarrow w_{jk} + \Delta w_{jk}$$
$$\Delta w_{jk} = \eta \sum_{i=1}^m h(x_i)(1-h(x_i))(d_i - h(x_i))x_{ijk}$$

Note this is similar to the previous update rule except for the extra term $h(x_i)(1-h(x_i))$, derivation of the sigmoid function

Minimum Description Length hypotheses

- Occam's razor
 - Choose the shortest explanation for the observed data

$$h_{MAP} = \arg \max_{h \in H} P(D | h)P(h)$$

$$h_{MAP} = \arg \max_{h \in H} \log_2 P(D | h) + \log_2 P(h)$$

$$h_{MAP} = \arg \min_{h \in H} -\log_2 P(D | h) - \log_2 P(h)$$

- Short hypotheses are preferred

Minimum Description Length hypotheses (cont.)

- From coding theory

$$L_{C_H}(h) = -\log_2 P(h)$$

where C_H is the optimal encoding for H

$$L_{C_{D|h}}(D|h) = -\log_2 P(D|h)$$

where $C_{D|h}$ is the optimal encoding for D given h

$$h_{MAP} = \arg \min_{h \in H} L_{C_H}(h) + L_{C_{D|h}}(D|h)$$

h_{MAP} minimizes the sum given by the description length of the hypothesis plus the description length of data

Minimum Description Length hypotheses (cont.)

MDL principle : choose h_{MDL} where

$$h_{MDL} = \arg \min_{h \in H} L_{C_1}(h) + L_{C_2}(D | h)$$

- C_1 : codes used to represent the hypothesis
- C_2 : codes used to represent the data given the hypothesis

If C_1 is chosen to be the optimal encoding of hypothesis C_H and C_2 to be the optimal encoding of hypothesis $C_{D|h}$, then $h_{MDL} = h_{MAP}$

Minimum Description Length hypotheses (cont.)

- Problem of learning decision trees
 - C_1 : Encoding of decision tree, in which the description length grows with the number of nodes and edges
 - C_2 : Encoding of data given a particular decision tree hypothesis in which description length is the number of bits necessary for identifying misclassification by the hypothesis.
 - No error in hypothesis classification : zero bit
 - Some error in hypothesis classification : at most $(\log_2 m + \log_2 k)$ bits when m is the number of training examples and k is the number of possible classifications
- MDL principle provides a way of trading off hypothesis complexity for the number of errors committed by the hypothesis
 - One method for dealing with the issue of over-fitting the data

Bayes Optimal Classification

- “What is *the most probable hypothesis* given the training data?”
=> “What is *the most probable classification* of the new instance given the training data?”
 - *MAP* hypothesis may be simply applied to the new instance
- Bayes optimal classification

$$\arg \max_{v_j \in V} \sum_{h_i \in H} P(v_j | h_i) P(h_i | D)$$

- Most probable classification of the new instance is obtained by combining the predictions of all hypothesis, weighted by their posterior probabilities

Bayes Optimal Classification (cont.)

- Example

- Posterior hypothesis

- $h_1=0.4, h_2=0.3, h_3=0.3$

- A set of possible classifications of the new instance is $V = \{ +, - \}$

- $P(h_1|D)=.4 \quad P(-|h_1)=0 \quad P(+|h_1)=1$

- $P(h_2|D)=.3 \quad P(-|h_2)=1 \quad P(+|h_2)=0$

- $P(h_3|D)=.3 \quad P(-|h_3)=1 \quad P(+|h_3)=0$

- $$\sum_{h_i \in H} P(+|h_i)P(h_i|D) = .4$$

- $$\sum_{h_i \in H} P(-|h_i)P(h_i|D) = .6$$

Bayes Optimal Classification (cont.)

$$\arg \max_{v_j \in \{+, -\}} \sum_{h_i \in H} P(v_j | h_i) P(h_i | D) = -$$

- This method maximizes the probability that the new instance is classified correctly (No other classification method using the same hypothesis space and same prior knowledge can outperform this method on average.)
- Example
 - In learning boolean concepts using version spaces, the Bayes optimal classification of a new instance is obtained by taking a weighted vote among all members of the version spaces, with each candidate hypothesis weighted by its posterior probability
- Note that the predictions it makes can correspond to a hypothesis not contained in H

Gibbs Algorithm

- Bayes optimal classifier
 - Obtains the best performance, given training data
 - Can be quite costly to apply
- Gibbs algorithm
 - Chooses a hypothesis h from H at random, according to the posterior probability distribution over H , $P(h|D)$, and uses h to predict the classification of the next instance x
 - Under certain conditions the expected error is at most twice those of the Bayes optimal classifier (Harssler et al. 1994)

Optimal Bayes Classifier

- Let each instance x be described by a conjunction of attribute values, where the target function $f(x)$ can take on any value from some finite set V
- Bayesian approach to classifying the new instance
 - Assigns the most probable target value

$$v_{MAP} = \arg \max_{v_j \in V} P(v_j | a_1, a_2 \dots a_n)$$

Optimal Bayes Classifier (cont.)

- Rewrite with Bayes theorem

$$\begin{aligned}v_{MAP} &= \arg \max_{v_j \in V} \frac{P(a_1, a_2 \dots a_n | v_j) P(v_j)}{P(a_1, a_2 \dots a_n)} \\ &= \arg \max_{v_j \in V} P(a_1, a_2 \dots a_n | v_j) P(v_j)\end{aligned}$$

- How to estimate $P(a_1, a_2, \dots, a_n | v_j)$ and $P(v_j)$?
(Not feasible unless a set of training data is very large but the number of different $P(a_1, a_2, \dots, a_n | v_j) =$ the number of possible instances \times the number of possible target values.)
- Hypothesis space
 $= \{ \langle P(v_j), P(\langle a_1, a_2, \dots, a_n \rangle | v_j) \rangle \mid (v_j \in V) \text{ and } (\langle a_1, a_2, \dots, a_n \rangle \in A_1 \times A_2 \times \dots \times A_n) \}$

Naive Bayes Classifier

- Assume
 - Attribute values are conditionally independent given the target value

$$P(a_1, a_2 \dots a_n | v_j) = \prod_i P(a_i | v_j)$$

- Naive Bayes classifier

$$v_{NB} = \arg \max_{v_j \in V} P(v_j) \prod_i P(a_i | v_j)$$

- Hypothesis space

$$= \{ \langle p(v_j), p(a_1/v_j), \dots, p(a_n/v_j) \rangle \mid (v_j \in V) \text{ and } (\langle a_i \in A_i, i=1, \dots, n \rangle) \}$$

- NB classifier needs a learning step to estimate its hypothesis space from the training data.

- If the naive Bayes assumption of conditional independence is satisfied, then, $v_{NB} = v_{MAP}$

Naive Bayes Classifier (cont.)

- An illustrative example
 - *PlayTennis* problem
 - Table 3.2 from Chapter 3
 - Classify the new instance
 - $\langle \text{Outlook} = \text{sunny}, \text{Temperature} = \text{cool}, \text{Humidity} = \text{high}, \text{Wind} = \text{strong} \rangle$
 - Predict target value (yes or no)

$$\begin{aligned}v_{NB} &= \arg \max_{v_j \in \{\text{yes}, \text{no}\}} P(v_j) \prod_i P(a_i | v_j) \\ &= \arg \max_{v_j \in \{\text{yes}, \text{no}\}} P(v_j) P(\text{Outlook} = \text{sunny} | v_j) P(\text{Temperature} = \text{cool} | v_j) \\ &\quad P(\text{Humidity} = \text{high} | v_j) P(\text{Wind} = \text{strong} | v_j)\end{aligned}$$

Naive Bayes Classifier (cont.)

- From training examples
 - Probabilities of the different target values
 - $P(\text{Play Tennis} = \text{yes}) = 9/14 = .64$
 - $P(\text{Play Tennis} = \text{no}) = 5/14 = .36$
 - ...
 - the conditional probabilities
 - $P(\text{Wind} = \text{strong} \mid \text{Play Tennis} = \text{yes}) = 3/9 = .33$
 - $P(\text{Wind} = \text{strong} \mid \text{Play Tennis} = \text{no}) = 3/5 = .60$
 - ...
 - $P(\text{yes})P(\text{sunny}/\text{yes})P(\text{cool}/\text{yes})P(\text{high}/\text{yes})P(\text{strong}/\text{yes}) = .0053$
 - $P(\text{no}) P(\text{sunny}/\text{no}) P(\text{cool}/\text{no}) P(\text{high}/\text{no}) P(\text{strong}/\text{no}) = .0206$

Naive Bayes Classifier (cont.)

- Target value, *PlayTennis*, of this new instance is *no*.
- Conditional probability that target value is *no* is

$$\frac{.0206}{.0206 + .0053} = .795$$

Naive Bayes Classifier (cont.)

- Estimating probabilities

- Conditional probability estimation by $\frac{n_c}{n}$

- poor when n_c is very small

- m -estimate of probability

$$\frac{n_c + mp}{n + m}$$

p : prior estimate of probability we wish to determine from n_c/n

m : a constant (equivalent sample size)

- Can be interpreted as augmenting the n actual observations by an additional m virtual samples distributed according to p

- Example : Let $p(\text{wind}=\text{strong} \mid \text{playtennis}=\text{no}) = 0.08$

If ‘*wind*’ has k possible values, then $p=1/k$ is assumed.

Example: Learning to Classify Text

- Instance space X : all possible text documents
Target value : { *like*, *dislike* }
- Design issues involved in applying the naive Bayes classifier
 - Represent an arbitrary text document in terms of attribute values
 - Estimate the probabilities required by the naive Bayes classifier

Example: Learning to Classify Text (cont.)

- Represent arbitrary text documents
 - an attribute - each word position in the document
 - the value of that attribute - the English word found in that position
- For the new text document (p180),

$$\begin{aligned}v_{NB} &= \arg \max_{v_j \in \{like, dislike\}} P(v_j) \prod_{i=1}^{111} P(a_i | v_j) \\ &= \arg \max_{v_j \in \{like, dislike\}} P(v_j) P(a_1 = "our" | v_j) P(a_2 = "approach" | v_j) \\ &\quad \dots P(a_{111} = "trouble" | v_j)\end{aligned}$$

Example: Learning to Classify Text (cont.)

- The independence assumption

$$P(a_1, \dots, a_{111} | v_j) = \prod_{i=1}^{111} P(a_i | v_j)$$

- The word probability for one text position are independent of the words that occur in other positions, given the document classification v_j
- Clearly incorrect
ex) “machine” and “learning”
- Fortunately, in practice the naive Bayes learner performs remarkably well in many text classification problems despite the incorrectness of this independence assumption

Example: Learning to Classify Text (cont.)

- $P(v_j)$
 - Can easily be estimated based on the fraction of each class in the training data
 - $P(\text{like}) = .3$ $P(\text{dislike}) = .7$
- $P(a_i | v_j)$
 - Must estimate a probability term for each combination of text position, English word, and target value
 - ➔ about 10 million such terms
 - Assume : the probability of encountering a specific word w_k is independent of the specific word position being considered

Example: Learning to Classify Text (cont.)

$$P(a_i = w_k | v_j) = P(a_m = w_k | v_j) \quad \text{for all } i, j, k, \text{ and } m$$

- Estimate the entire set of probability by the single position-independent probability $P(w_k | v_j)$

- Estimate : Adopt the m -estimate

$$P(w_k | v_j) = \frac{n_c + mp}{n + m} = \frac{n_k + 1}{n + |\text{Vocabulary}|}$$

- Document classification

$$v_{NB} = \arg \max_{v_j \in V} P(v_j) \prod_{i \in \text{positions}} P(a_i | v_j)$$

Example: Learning to Classify Text (cont.)

- Experimental result
 - Classify usenet news article (Joachims, 1996)
 - 20 possible newsgroups
 - 1,000 articles were collected per each group
 - Use 2/3 of 20,000 docs as training examples
 - Performance was measured over the remaining 1/3
 - The accuracy achieved by the program was 89%

Bayesian Belief Networks

- Naive Bayes classifier
 - Assumption of conditional independence of the attributes → simple but too restrictive → intermediate approach
- Bayesian belief networks
 - Describes the probability distribution over a set of variables by specifying conditional independence assumptions with a set of conditional probabilities.
 - Joint space: $V(Y_1) \times V(Y_2) \times \dots \times V(Y_n)$
 - Joint probability distribution : Probability for each of the possible bindings for the tuple $\langle Y_1 \dots Y_n \rangle$

Bayesian Belief Networks (cont.)

- Conditional Independence

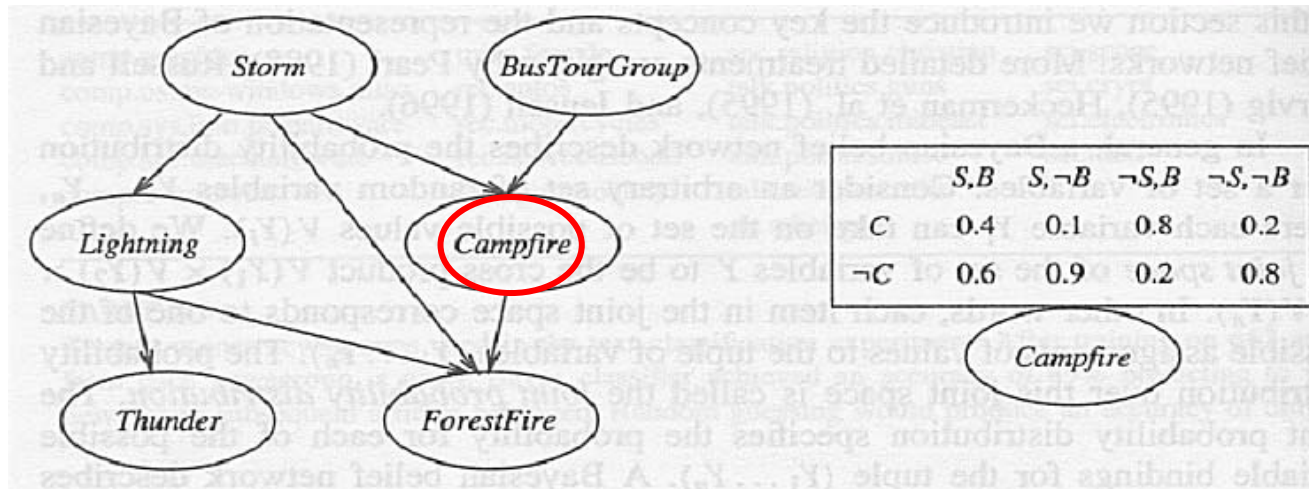
- X is conditionally independent of Y given Z : When the probability distribution governing X is independent of the value of Y given a value Z
- Extended form:

$$P(X_1 \dots X_l | Y_1 \dots Y_m, Z_1 \dots Z_n) = P(X_1 \dots X_l | Z_1 \dots Z_n)$$

Bayesian Belief Networks (cont.)

- Representation

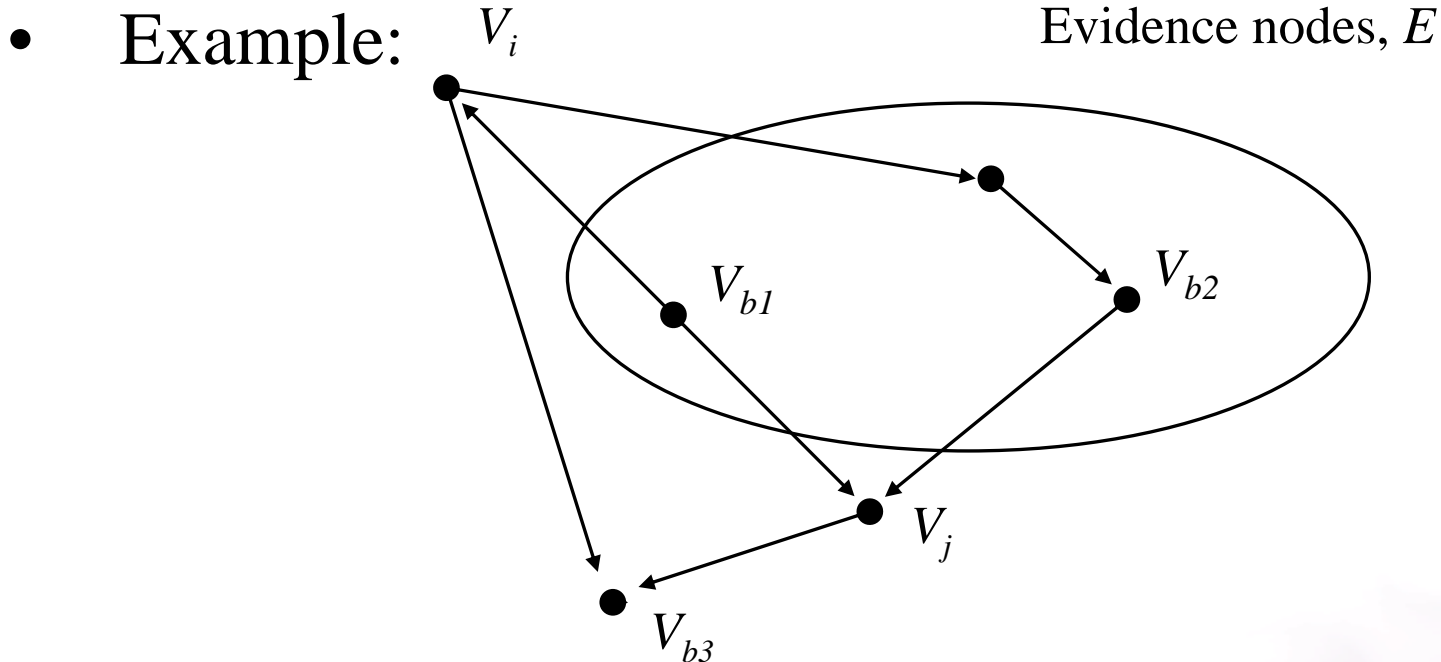
- Directed acyclic graph
- Node : each variable
- For each variable next two are given
 - Network arcs: variable is conditionally independent of its nondescendants in the network given its immediate predecessors
 - Conditional probability table (Hypothesis space)
- D-separation (conditional dependency in the network)



Bayesian Belief Networks (cont.)

- D-separation (conditional dependency in the network)
 - Two nodes V_i and V_j are conditionally independent given a set of nodes ε (that is $I(V_i, V_j | \varepsilon)$) if for every undirected path in the Bayes network between V_i and V_j , there is some node, V_b , on the path having one of the following three properties
 1. V_b is in ε , and both arcs on the path lead out of V_b
 2. V_b is in ε , and one arc on the path leads in to V_b and one arc leads out.
 3. Neither V_b nor any descendant of V_b is in ε , and both arcs on the path lead in to V_b .

Bayesian Belief Networks (cont.)



V_i is independent of V_j given the evidence nodes because all three paths between them are blocked. The blocking nodes are

- (a) V_{b1} is an evidence node, and both arcs lead out of V_{b1} .
- (b) V_{b2} is an evidence node, and one arc leads into V_{b2} and one arc leads out.
- (c) V_{b3} is not an evidence node, nor are any of its descendants, and both arcs lead into V_{b3} .

Bayesian Belief Networks (cont.)

- The joint probability for any desired assignment of values $\langle y_1, \dots, y_n \rangle$ to the tuple of network variables $\langle Y_1, \dots, Y_n \rangle$

$$P(y_1, \dots, y_n) = \prod_{i=1}^n P(y_i \mid \text{Parents}(Y_i))$$

- The values of $P(y_i \mid \text{Parents}(Y_i))$ are precisely the values stored in the conditional probability table associated with node Y_i

Bayesian Belief Networks (cont.)

- Inference

- Infer the value of some target variable, given the observed values of the other variables
- More accurately, infer the probability distribution for target variable, specifying the probability that it will take on each of its possible values given the observed values of the other variables.
 - Example : Let the Bayesian belief network with $(n+1)$ attributes (variables) A_1, \dots, A_n, T , be constructed from the training data.

Then the target value of the new instance $\langle a_1, \dots, a_n \rangle$ would be

$$\arg \max_{v_i \in V(T)} P(T = v_i \mid A_1 = a_1, \dots, A_n = a_n)$$

- Exact inference of probabilities \rightarrow generally, NP-hard
 - Monte Carlo methods : Approximate solutions by randomly sampling the distributions of the unobserved variables
 - Polytree network : Directed acyclic graph in which there is just one path, along edges in either direction, between only two nodes.

Bayesian Belief Networks (cont.)

- Learning Bayesian belief networks
 - Different settings of learning problem
 - Network structure: known
 - Case 1: all variables observable → Straightforward
 - Case 2: some variables observable → Gradient ascent procedure
 - Network structure: unknown
 - Bayesian scoring metric
 - K2

Bayesian Belief Networks (cont.)

- Gradient ascent training of B.B.N.
 - Structure is known, variables are partially observable
 - Similar to learn the weights for the hidden units in an neural network
 - Goal : Find $h_{ML} \equiv \arg \max_{h \in H} P(D | h)$
 - Use of a gradient ascent method

Bayesian Belief Networks (cont.)

- Maximizes $P(D | h)$ by following the gradient of $\ln P(D | h)$

$$\frac{\partial \ln P(D | h)}{\partial w_{ijk}} = \sum_{d \in D} \frac{P(Y_i = y_{ij}, U_{ij} = u_{ik} | d)}{w_{ijk}}$$

- Y_i : network variable
- U_i : $Parents(Y_i)$
- w_{ijk} : a single entry in conditional probability table
- $w_{ijk} = P(Y_i = y_{ij} | U_i = u_{ik})$
- For example, if Y_i is the variable *Campfire*, then $y_{ij} = True$, $u_{ik} = \langle False, False \rangle$

Bayesian Belief Networks (cont.)

- Perform gradient ascent repeatedly

1. Update w_{ijk} using D

$$w_{ijk} \leftarrow w_{ijk} + \eta \sum_{d \in D} \frac{P_h(y_{ij}, u_{ik} | d)}{w_{ijk}}$$

η : learning rate

2. Renormalize w_{ijk} to assure $\sum_j w_{ijk} = 1$, $0 \leq w_{ijk} \leq 1$

Bayesian Belief Networks (cont.)

- Derivation process of $\frac{\partial \ln P_h(D)}{\partial w_{ijk}}$

– Assume that the training example d in the data set D are drawn independently

$$\begin{aligned}\frac{\partial \ln P_h(D)}{\partial w_{ijk}} &= \frac{\partial}{\partial w_{ijk}} \ln \prod_{d \in D} P_h(d) \\ &= \sum_{d \in D} \frac{\partial \ln P_h(d)}{\partial w_{ijk}} \\ &= \sum_{d \in D} \frac{1}{P_h(d)} \frac{\partial P_h(d)}{\partial w_{ijk}}\end{aligned}$$

$$\begin{aligned}\frac{\partial \ln P_h(D)}{\partial w_{ijk}} &= \sum_{d \in D} \frac{1}{P_h(d)} \frac{\partial}{\partial w_{ijk}} \sum_{j',k'} P_h(d | y_{ij'}, u_{ik'}) P_h(y_{ij'}, u_{ik'}) \\ &= \sum_{d \in D} \frac{1}{P_h(d)} \frac{\partial}{\partial w_{ijk}} \sum_{j',k'} P_h(d | y_{ij'}, u_{ik'}) P_h(y_{ij'} | u_{ik'}) P(u_{ik'})\end{aligned}$$

Bayesian Belief Networks (cont.)

- Given that $w_{ijk} \equiv P_h(y_{ij} | u_{ik})$, the only term in this sum for which $\frac{\partial}{\partial w_{ijk}}$ is nonzero is the term for which $j' = j$ and $i' = i$

$$\begin{aligned}\frac{\partial \ln P_h(D)}{\partial w_{ijk}} &= \sum_{d \in D} \frac{1}{P_h(d)} \frac{\partial}{\partial w_{ijk}} P_h(d | y_{ij}, u_{ik}) P_h(y_{ij} | u_{ik}) P(u_{ik}) \\ &= \sum_{d \in D} \frac{1}{P_h(d)} \frac{\partial}{\partial w_{ijk}} P_h(d | y_{ij}, u_{ik}) w_{ijk} P(u_{ik}) \\ &= \sum_{d \in D} \frac{1}{P_h(d)} P_h(d | y_{ij}, u_{ik}) P(u_{ik})\end{aligned}$$

Bayesian Belief Networks (cont.)

- Applying Bayes theorem,

$$\begin{aligned}\frac{\partial \ln P_h(D)}{\partial w_{ijk}} &= \sum_{d \in D} \frac{1}{P_h(d)} \frac{P_h(y_{ij}, u_{ik} | d) P_h(d) P_h(u_{ik})}{P_h(y_{ij}, u_{ik})} \\ &= \sum_{d \in D} \frac{P_h(y_{ij}, u_{ik} | d) P(u_{ik})}{P_h(y_{ij}, u_{ik})} \\ &= \sum_{d \in D} \frac{P_h(y_{ij}, u_{ik} | d)}{P_h(y_{ij} | u_{ik})} \\ &= \sum_{d \in D} \frac{P_h(y_{ij}, u_{ik} | d)}{w_{ijk}}\end{aligned}$$

Bayesian Belief Networks (cont.)

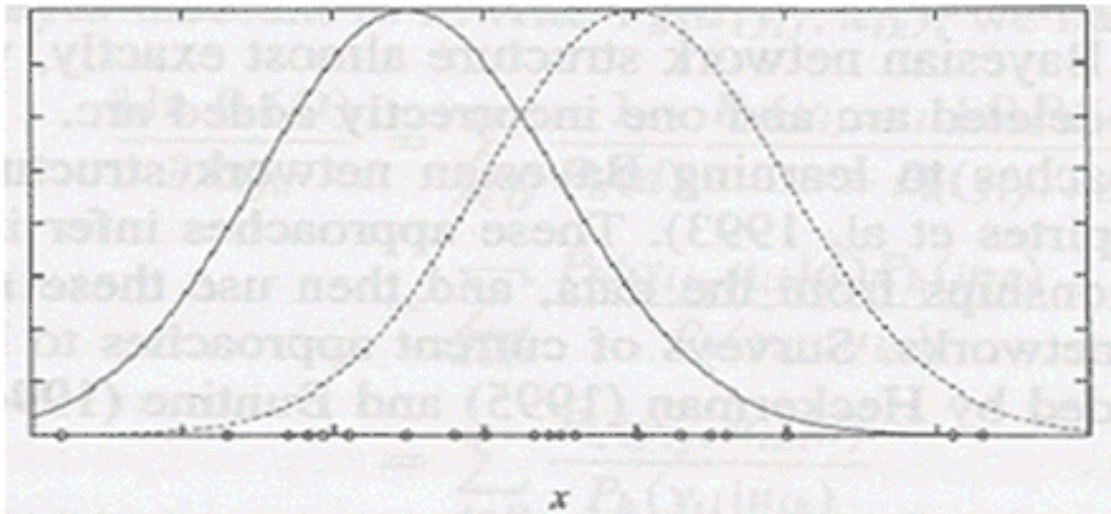
- Learning the structure of Bayesian networks
 - Bayesian scoring metric (Cooper and Herskovits, 1992)
 - K2 algorithm
 - Heuristic greedy search algorithm when data is fully observed data
 - Constraint-based approach (Spirtes *et al*, 1993)
 - Infer dependency and independency relationships from data
 - Construct structure using this relationship

EM Algorithm

- When to use
 - Learning in the presence of unobserved variables
 - When the form of probability distribution is known
- Applications
 - Training Bayesian belief networks
 - Training radial basis function networks (Chapter 8.)
 - Basis of many unsupervised clustering algorithms

EM Algorithm (cont.)

- Estimating means of k Gaussians
 - Each instance is generated using a two-steps
 1. Select one of the k normal distributions at random
(all the σ s of the distributions are the same and known)
 2. Generate an instance x_i according to this selected distribution



EM Algorithm (cont.)

- Task
 - Finding (maximum likelihood) hypothesis $h = \langle \mu_1, \dots, \mu_k \rangle$, that maximizes $p(D/h)$
- Conditions
 - Instances from X are generated by mixture of k normal distributions.
 - Which x_i is generated by which distribution is unknown
 - Means of that k distribution, $\langle \mu_1, \dots, \mu_k \rangle$, are unknown

EM Algorithm (cont.)

- Single normal distribution

$$\mu_{ML} = \arg \min \sum_i (x_i - \mu)^2 = \frac{1}{m} \sum_i x_i$$

- Two normal distribution

$$y_i = \langle x_i, z_{i1}, z_{i2} \rangle$$

$$z_{ij} = \begin{cases} 1 & \text{if } x_i \text{ generated by } j \text{ th distribution} \\ 0 & \text{otherwise} \end{cases}$$

x_j : i th observed value

- If z is known : Use the straightforward way
- Else use EM algorithm : Repeated re-estimating

EM Algorithm (cont.)

- Initialize $h = \langle \mu_1, \mu_2 \rangle$ with arbitrary small value
- Step 1 : Calculate $E[z_{ij}]$, assuming h holds

$$E[z_{ij}] = \frac{p(x_i | \mu_j)}{\sum_{n=1}^2 p(x_i | \mu_n)} = \frac{e^{-\frac{1}{2\sigma^2}(x_i - \mu_j)^2}}{\sum_{n=1}^2 e^{-\frac{1}{2\sigma^2}(x_i - \mu_n)^2}}$$

- Step 2 : Calculate a new maximum likelihood hypothesis $h' = \langle \mu_1', \mu_2' \rangle$ (use $E[z_{ij}]$ from step1)

$$\mu_j \leftarrow \frac{1}{m} \sum_{i=1}^m E[z_{ij}] x_i$$

- Until the procedure converges to a stationary value for h

EM Algorithm (cont.)

- General statement of EM algorithm
 - Given:
 - Observed data $X = \{x_1, \dots, x_n\}$
 - Unobserved data $Z = \{z_1, \dots, z_n\}$
 - Parameterized probability distribution $P(Y|h)$, where
 - $Y = \{y_1, \dots, y_n\}$ is the full data $y_i = x_i \cup z_i$
 - θ : underlying probability distribution
 - h : current hypothesis of θ
 - h' : revised hypothesis
 - Determine:
 - h that (locally) maximizes $E[\ln P(Y|h)]$

EM Algorithm (cont.)

- Assume $\theta = h$, define $Q(h' | h)$

$$Q(h' | h) = E[\ln p(Y | h') | h, X]$$

- Repeats until convergence:
 - Step 1: Estimation step:

$$Q(h' | h) \leftarrow E[\ln p(Y | h') | h, X]$$

- Step 2: Maximization step:

$$h \leftarrow \operatorname{argmax}_{h'} Q(h' | h)$$

EM Algorithm (cont.)

- Example : Derivation of the k mean algorithm

- The probability $p(y_i | h')$ of a single instance $y_i = \langle x_i, z_{i1}, \dots, z_{ik} \rangle$ of the full data

$$p(y_i | h') = p(x_i, z_{i1}, z_{i2}, \dots, z_{ik} | h') = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2\sigma^2} \sum_j^k z_{ij} (x_i - \mu'_j)^2}$$

- only one of z_{ij} can have the value 1, and all others must be 0

- $\ln P(Y | h') = \ln \prod_{i=1}^m p(y_i | h')$

$$= \sum_{i=1}^m \ln p(y_i | h')$$

$$= \sum_{i=1}^m \left(\ln \frac{1}{\sqrt{2\pi\sigma^2}} - \frac{1}{2\sigma^2} \sum_{j=1}^k z_{ij} (x_i - \mu'_j)^2 \right)$$

EM Algorithm (cont.)

- Expression for $\ln P(y_i | h')$ is a linear function of these z_{ij}

$$\begin{aligned} E[\ln P(Y | h')] &= E \left[\sum_{i=1}^m \left(\ln \frac{1}{\sqrt{2\pi\sigma^2}} - \frac{1}{2\sigma^2} \sum_{j=1}^k z_{ij} (x_i - \mu'_j)^2 \right) \right] \\ &= \sum_{i=1}^m \left(\ln \frac{1}{\sqrt{2\pi\sigma^2}} - \frac{1}{2\sigma^2} \sum_{j=1}^k E[z_{ij}] (x_i - \mu'_j)^2 \right) \end{aligned}$$

- The function $Q(h' | h)$ for the k means problem

$$Q(h' | h) = \sum_{i=1}^m \left(\ln \frac{1}{\sqrt{2\pi\sigma^2}} - \frac{1}{2\sigma^2} \sum_{j=1}^k E[z_{ij}] (x_i - \mu'_j)^2 \right)$$

EM Algorithm (cont.)

$$E[z_{ij}] = \frac{e^{-\frac{1}{2\sigma^2}(x_i - \mu_j)^2}}{\sum_{n=1}^k e^{-\frac{1}{2\sigma^2}(x_i - \mu_n)^2}}$$

$$\begin{aligned} \arg \max_{h'} Q(h' | h) &= \arg \max_{h'} \sum_{i=1}^m \left(\ln \frac{1}{\sqrt{2\pi\sigma^2}} - \frac{1}{2\sigma^2} \sum_{j=1}^k E[z_{ij}] (x_i - \mu'_j)^2 \right) \\ &= \arg \min_{h'} \sum_{i=1}^m \sum_{j=1}^k E[z_{ij}] (x_i - \mu'_j)^2 \end{aligned}$$

Minimized by setting each μ'_j to the weighted sample mean

$$\mu_j \leftarrow \frac{1}{m} \sum_{i=1}^m E[z_{ij}] x_i$$